Pattern Search: A Status Report

Virginia Torczon
College of William & Mary

Collaborators:

- Liz Dolan, Argonne National Laboratory
- Adam Gurson, College of William & Mary
- Patty Hough, Sandia National Laboratory
- Tammy Kolda, Sandia National Laboratory
- Michael Lewis, College of William & Mary
- Anne Shepherd, College of William & Mary

Problem

$$\min_{x \in \mathcal{S}} f(x)$$

where $f : \mathbb{R}^n \to \mathbb{R}$ and $\mathcal{S} \subseteq \mathbb{R}^n$.

Pattern Search Methods

- Are designed to find (constrained) stationary points for optimization problems of this form.
- Must be adapted/modified based on the structure of the feasible region $\mathcal{S}$.

Engineering Example: Determining the Characteristics of a Circuit

- **Work by:** Tammy Kolda and Ken Marx (Sandia)
- **Variables:** inductances, capacitances, diode saturation currents, transistor gains, leakage inductances, and transformer core parameters
- **Simulation Code:** SPICE3

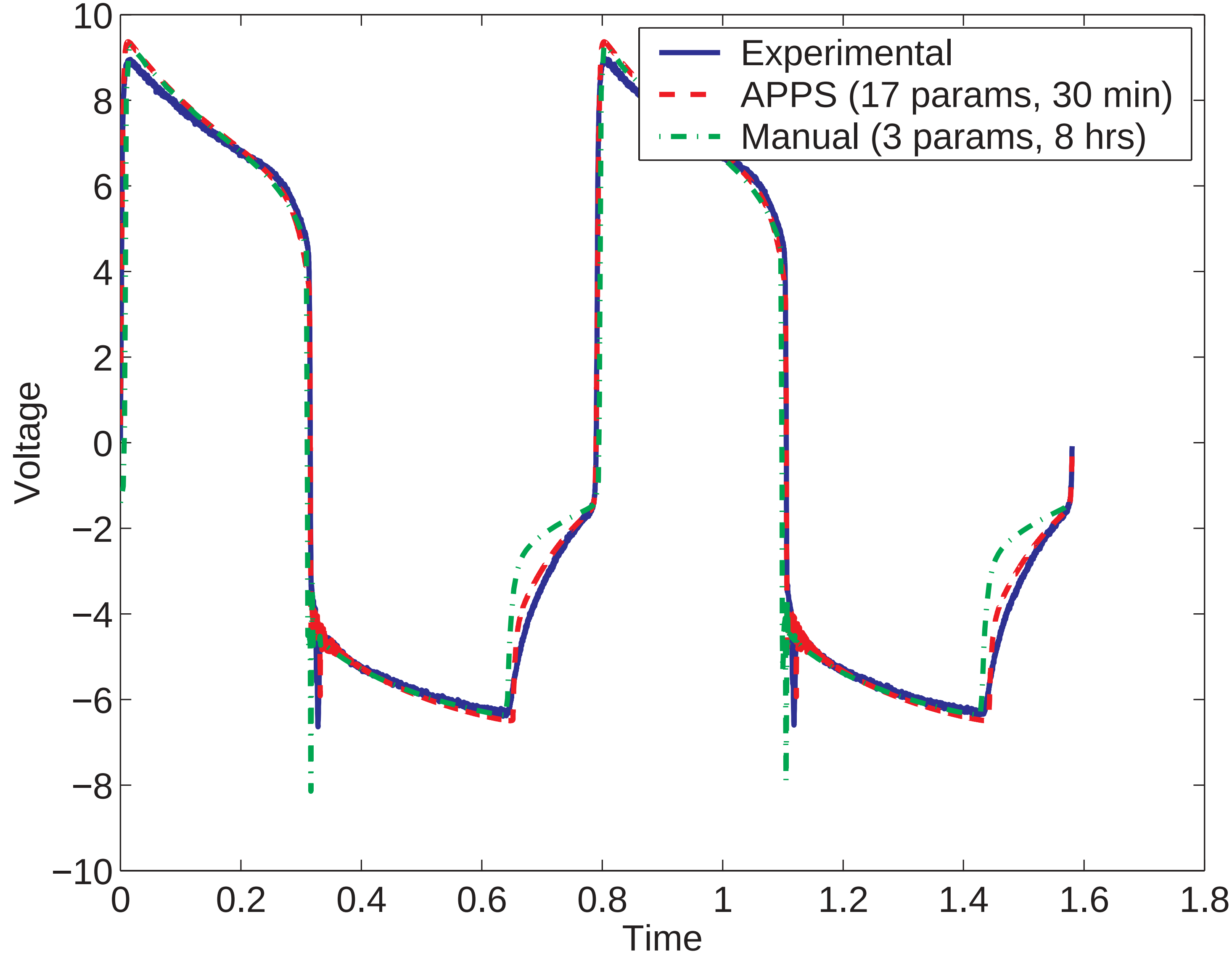
$$f(x) = \sum_{t=1}^{N} \left(V_t^{\mathrm{SIM}}(x) - V_t^{\mathrm{EXP}} \right)^2 ,$$

$$\begin{aligned} x &= \text{17 unknown characteristics} \\ V_t^{\mathrm{SIM}}(x) &= \text{Simulation voltage at time } t \\ V_t^{\mathrm{EXP}} &= \text{Experimental voltage at time } t \\ N &= \text{Number of time steps (2700)} \end{aligned}$$

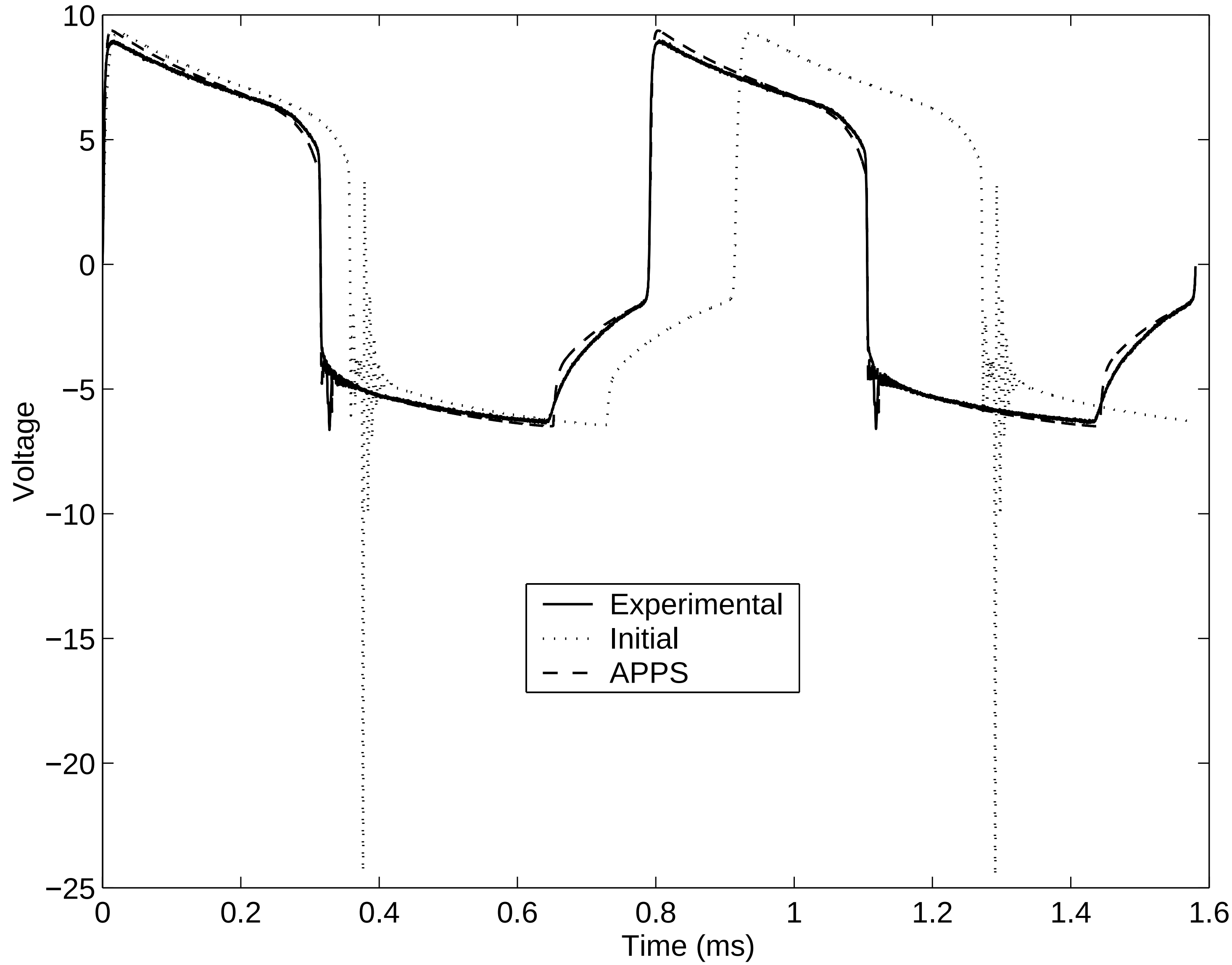
Why Use Pattern Search Methods?

- **No Derivatives:** Pattern search methods do not require derivative/gradient/sensitivity information.
- **Noisy Data:** Pattern search methods tend to do well even when the data is "noisy."
  (Assume the noise is *nonstochastic*: computer simulation always returns the same output for a given input, but the output is known to be inaccurate).
- **"Embarrassingly" Parallel:** Pattern search methods are amenable to parallelization, even when the objective (cost) function is serial.

APPS Solution is Faster and More Accurate than Hand-Tuning



APPS Solution Shows Significant Improvement over Initial Parameters



My Goals for This Talk

1. Describe what problems can be solved using pattern search methods.
   Two aspects: analysis and implementation.
2. Show *why* pattern search works.
3. Show where pattern search "fits" within the context of gradient-based optimization techniques.
4. Demonstrate that analysis is justified by the insight it provides.
5. Show where we are going next.

Metrics for "Success"

1. Can prove an algorithm converges to a (constrained) stationary point of the *original* problem:

$$\min_{x \in \mathcal{S}} f(x)$$

   where $f : \mathbb{R}^n \to \mathbb{R}$ and $\mathcal{S} \subseteq \mathbb{R}^n$.

2. Can envision robust implementations.

Bottom line: want effective implementations.

Ideally: the analysis opens up algorithmic possibilities and warns of potential pitfalls.

Assumptions on the Objective Function

Shared:

- f is nonlinear and
- f is (at least) continuously differentiable.

Specific to pattern search:

- ∇f is unavailable and
- approximations to ∇f are not reliable.

Structure of the Feasible Region

Shared:

- Unconstrained: $\mathcal{S} = \mathbb{R}^n$.
  "Easiest" to solve.
- Bound Constraints: $\ell \leq x \leq u$.
  "Easy" to solve.
- Linear Constraints: $\ell \leq Ax \leq u$.
  Possible to solve; active research area.
- General (Nonlinear) Constraints: $c(x) = 0$.
  Possible to solve; active research area.

Where Do We Stand with Pattern Search?

- Unconstrained
  - Analysis well in hand. [Berman, 1969; Polak, 1971; Céa, 1971; Wenci, 1979; Torczon, 1997; Lewis/Torczon, 1996; Kolda/Torczon, 2001]
  - Software available—sequential (`DirectSearch`) and distributed asynchronous (`APPS`). [Dolan, 1999; Hough/Kolda/Torczon, 1999; Gurson, 2000; Shepherd, 2001]
- Bound Constraints
  - Analysis well in hand. [Lewis/Torczon, 1999; Lewis/Torczon, 2000]
  - Software available—sequential (`DirectSearch`) only. [Dolan, 1999; Shepherd, 2001]

Where Do We Stand with Pattern Search? (cont.)

- Linear Constraints
  - Analysis well in hand. [Lewis/Torczon, 2000]
  - No available software.
    Need to handle degeneracy.
- General (Nonlinear) Constraints
  - Analysis in hand. [Lewis/Torczon, 2002]
    Should—will!—investigate other strategies.
  - No available software.

Executive Summary of Work Ahead

- Unconstrained
  - "Done."
- Bound Constraints
  - Asynchronous distributed implementation.
    (Fold in as a special case for linear constraints.)
- Linear Constraints
  - Sequential implementation for nondegenerate case.
  - Sequential implementation for degenerate case.
  - Asynchronous distributed implementation.
- General (Nonlinear) Constraints
  - More analysis investigating alternatives.
  - Sequential implementations.
  - Asynchronous distributed implementations.

The Analysis Provides the Key to Understanding

The analysis of *any* nonlinear optimization algorithm is based on the appropriate choice of:

1. search directions and
2. a step-length control mechanism
   —coupled with—
   a step acceptance criteria.

Gradient-Based vs. Pattern Search Methods

1. Search directions

   Gradient-Based: *single* search direction d.

   Pattern Search: *sufficient set* of directions $\mathcal{D}$, $|\mathcal{D}| \geq n + 1$, where "sufficient" is with respect to the cone of feasible directions.

2. Step-length control

   Gradient-Based: globalization strategies coupled with a *sufficient decrease* condition, where "sufficient" is with respect to the amount of decrease realized relative to the norm of the gradient.

   Pattern Search: all steps must lie on a rational lattice (a grid) coupled with a *simple decrease* condition.

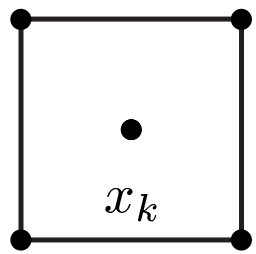
Why "Pattern" Search

Historical origins in the statistics literature on experimental design [G.E.P. Box, 1957]:

> define a *pattern* of points over which to sample the function.

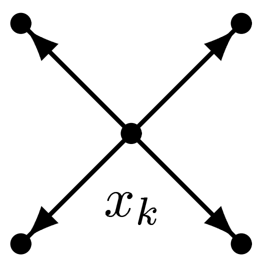
Examples

What the statistician sees:



two-level factorial design

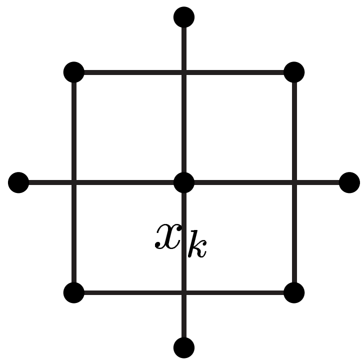
What the optimizer sees:



$$\left\{ \begin{pmatrix} 1 \\ 1 \end{pmatrix} \begin{pmatrix} 1 \\ -1 \end{pmatrix} \begin{pmatrix} -1 \\ 1 \end{pmatrix} \begin{pmatrix} -1 \\ -1 \end{pmatrix} \right\}$$

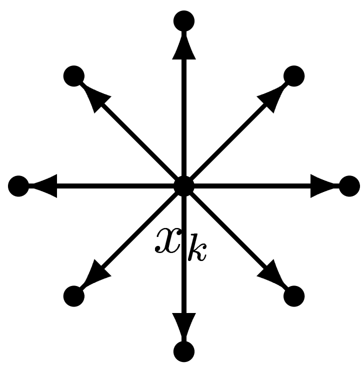
Examples (cont.)

What the statistician sees:



composite factorial design

What the optimizer sees:



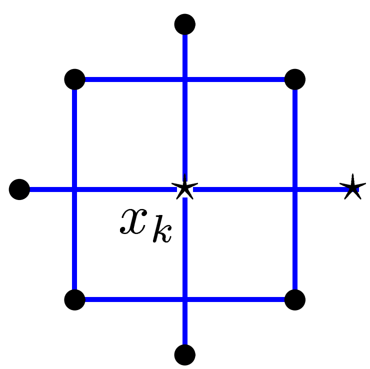
$$\left\{ \begin{pmatrix} 1 \\ 1 \end{pmatrix} \begin{pmatrix} 1 \\ -1 \end{pmatrix} \begin{pmatrix} -1 \\ 1 \end{pmatrix} \begin{pmatrix} -1 \\ -1 \end{pmatrix} \begin{pmatrix} 1.5 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1.5 \end{pmatrix} \begin{pmatrix} -1.5 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ -1.5 \end{pmatrix} \right\}$$

The Underlying Lattice

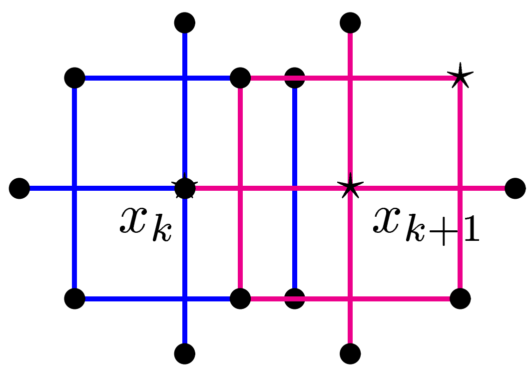
Because of the patterns, traditionally, most pattern search methods "naturally" restricted the steps to lattice points.

The unanticipated effect: a built-in step-length control mechanism.

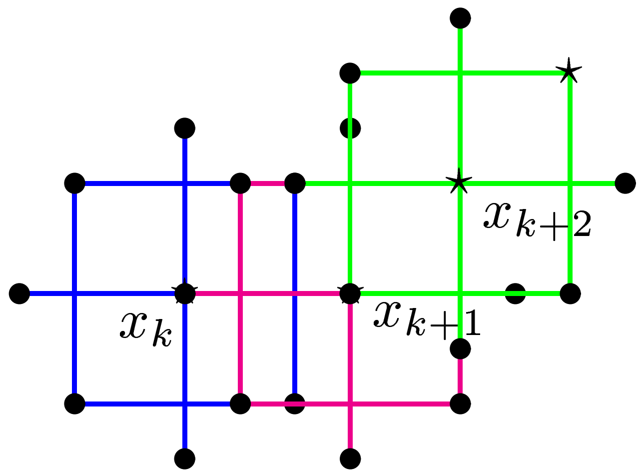
EXAMPLE: ITERATION k



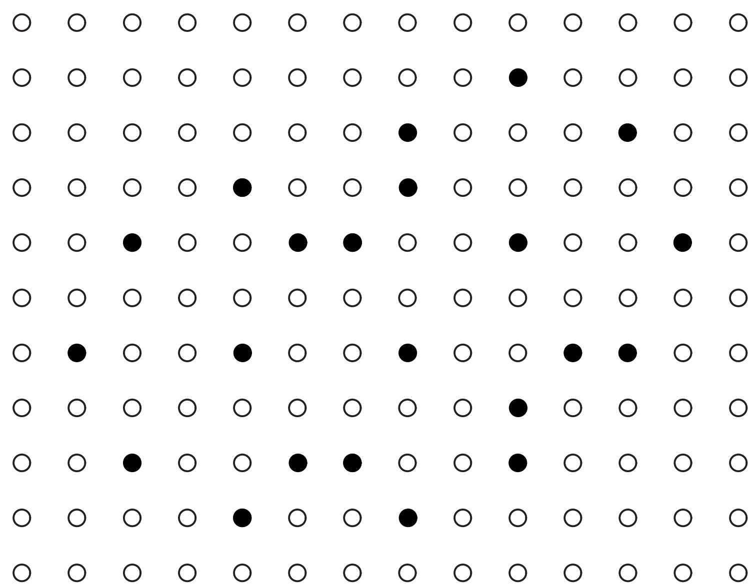
Example: Iteration $k + 1$



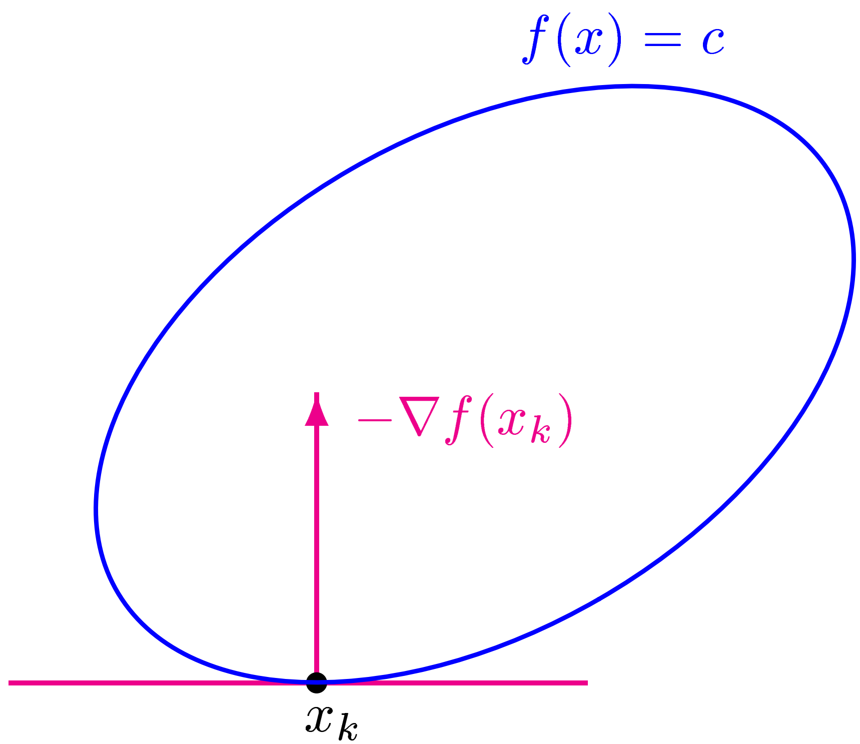
Example: Iteration $k+2$



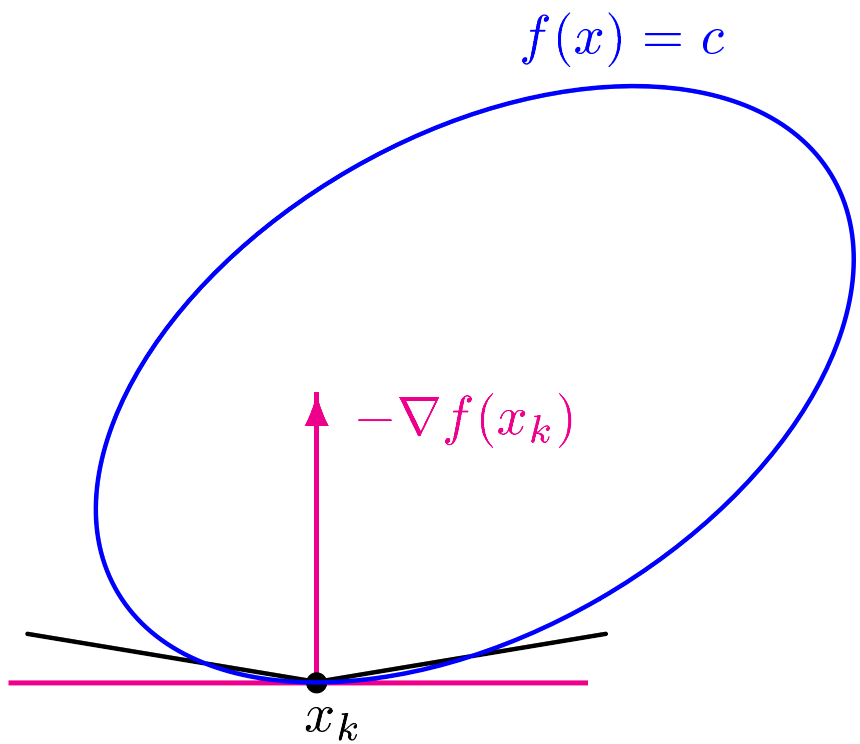
Example: Lattice Underlying the Search



Directions: Cone of Descent Directions

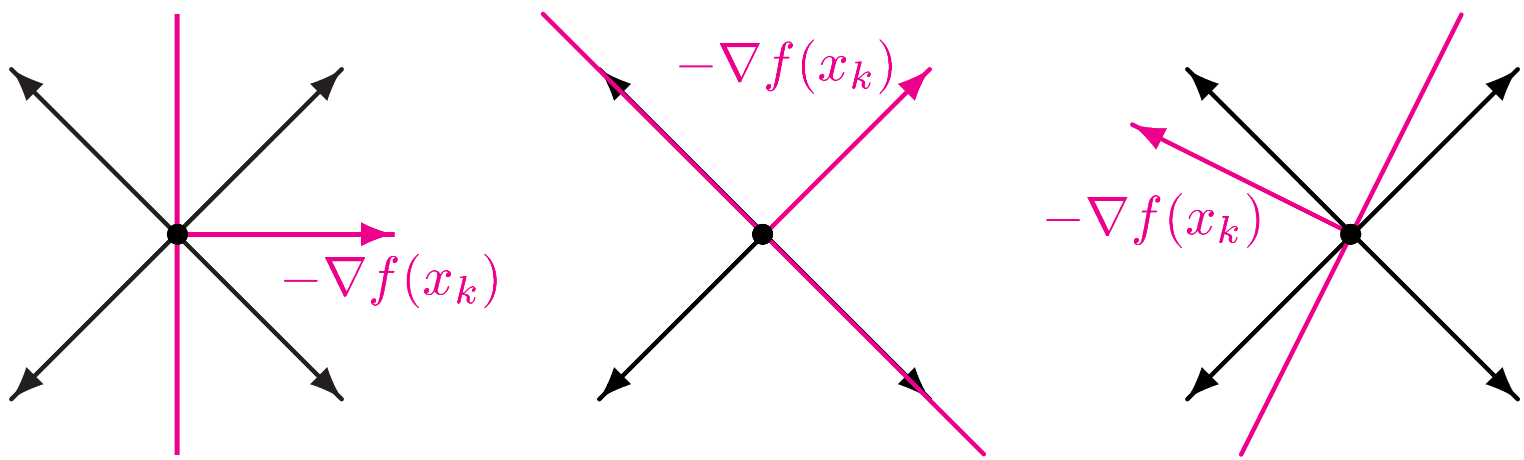


Directions: Require At Least One Search Direction in Cone



Directions: Pattern Search Guarantees At Least One in the Cone

IF f is continuously differentiable, then guaranteed at least one direction of descent, *even though we do not know* $-\nabla f(x_k)$.



Pattern search methods are gradient-related.

A Sufficient Set of Search Directions

A *sufficient* set of search directions should guarantee us that *if* f is differentiable, at least one search direction lies in the cone of descent directions.

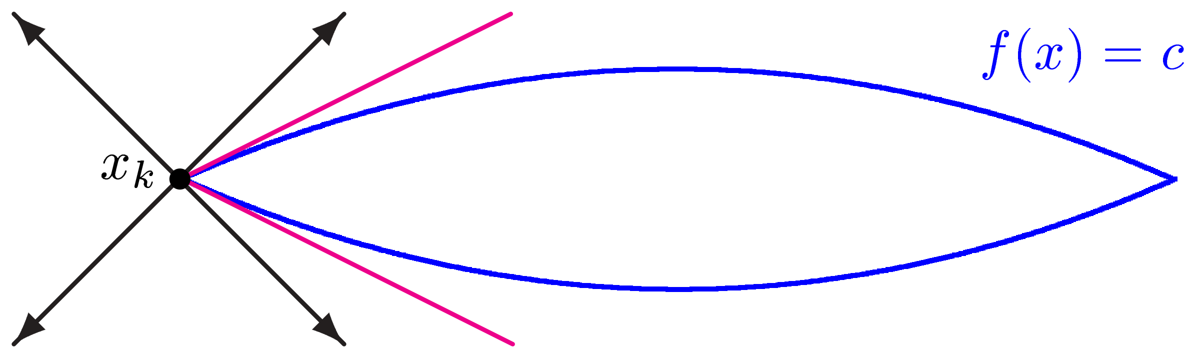
If the cone of descent directions is a half space—translated by x_k—then a *positive basis* satisfies our requirement by guaranteeing at least one direction of descent.

> **Positive Basis:** a set of vectors that allows us to write *any* vector in $\mathbb{R}^n$ as a nonnegative combination of the vectors in the positive basis.

For *unconstrained* optimization, the cone of descent directions *is* a half space translated by x_k.

Therefore a positive basis gives us a sufficient set of search directions for unconstrained minimization when f is differentiable.

The Differentiability of f is Critical



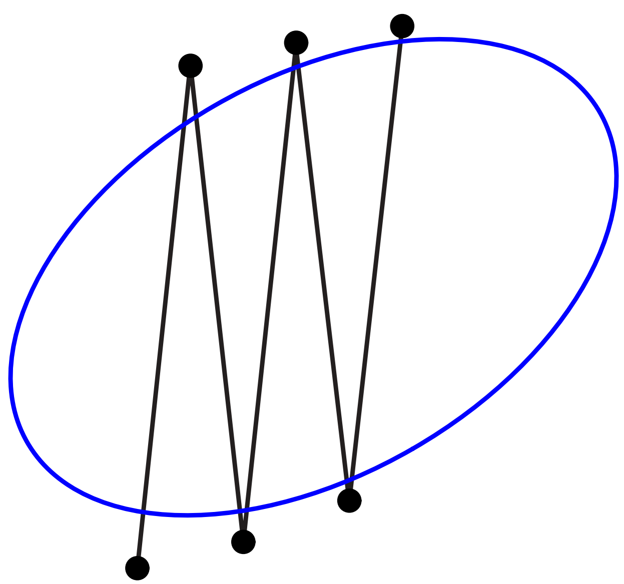
The cone of descent directions may not be a translated half space.

A positive basis is no longer guaranteed to form a sufficient set of search directions.

Common Outcome: convergence to a nonstationary point of f.

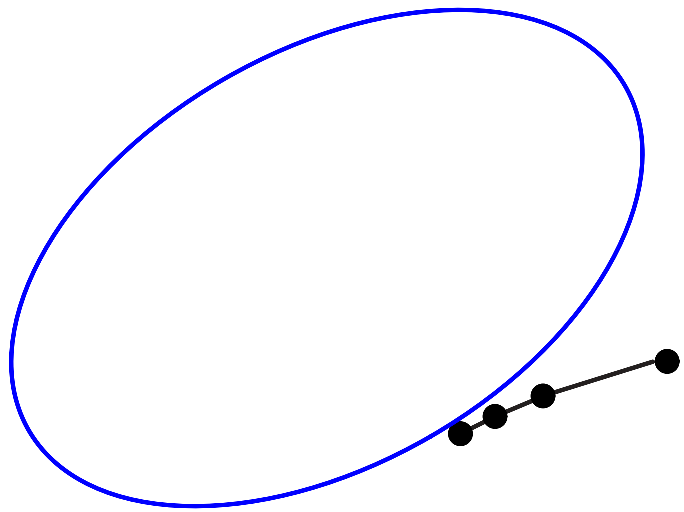
Any weakening of the assumption that f is continuously differentiable suffers from the same fundamental difficulty.

Lattice: Prevent Steps That Are "Too Long"



decrease too small
relative to the
length of the step

Lattice: Prevent Steps That Are "Too Short"

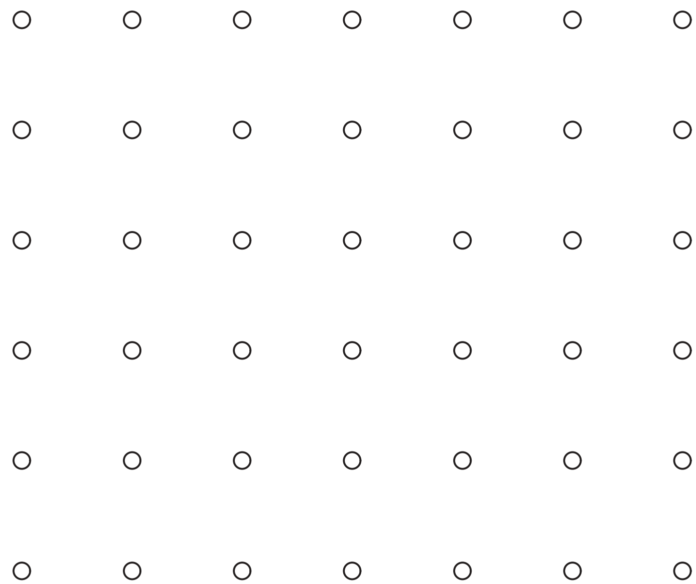


decrease too small

relative to the

norm of the gradient

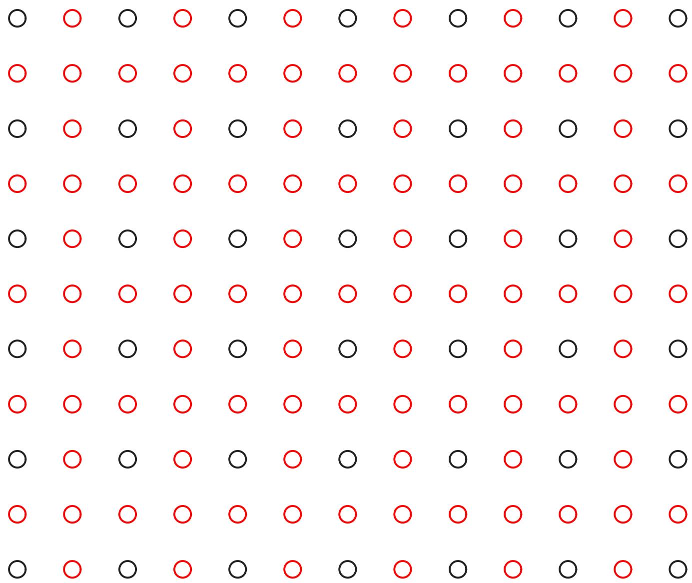
Lattice: Steps Cannot "Jam Up"



All steps must lie on the current lattice, so steps cannot become arbitrarily close.

Lattice: Refinement

Only when no more descent can be found using the current pattern.



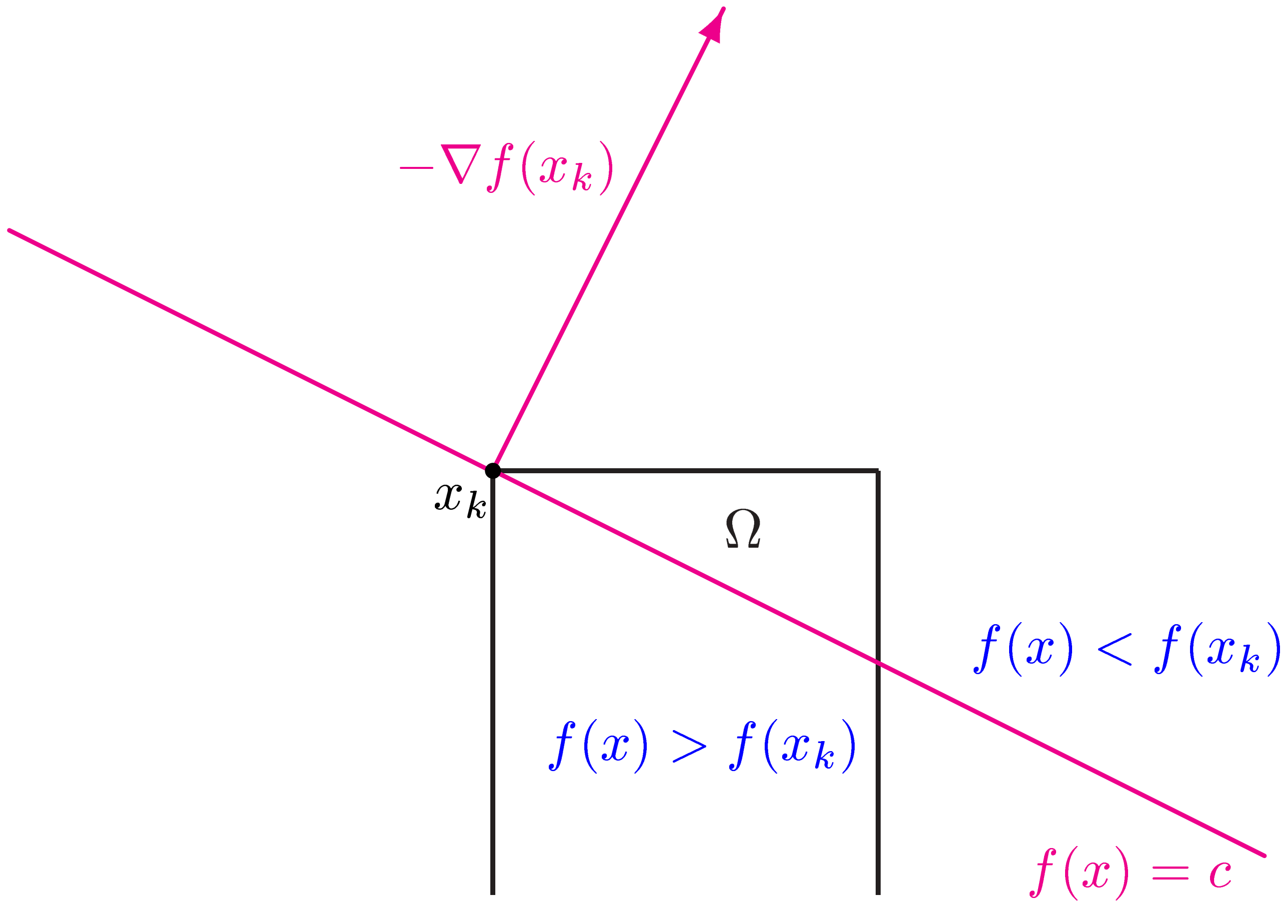
Search converges because the grid spacing goes to zero in the limit.

Summary of Pattern Search for the Unconstrained Case

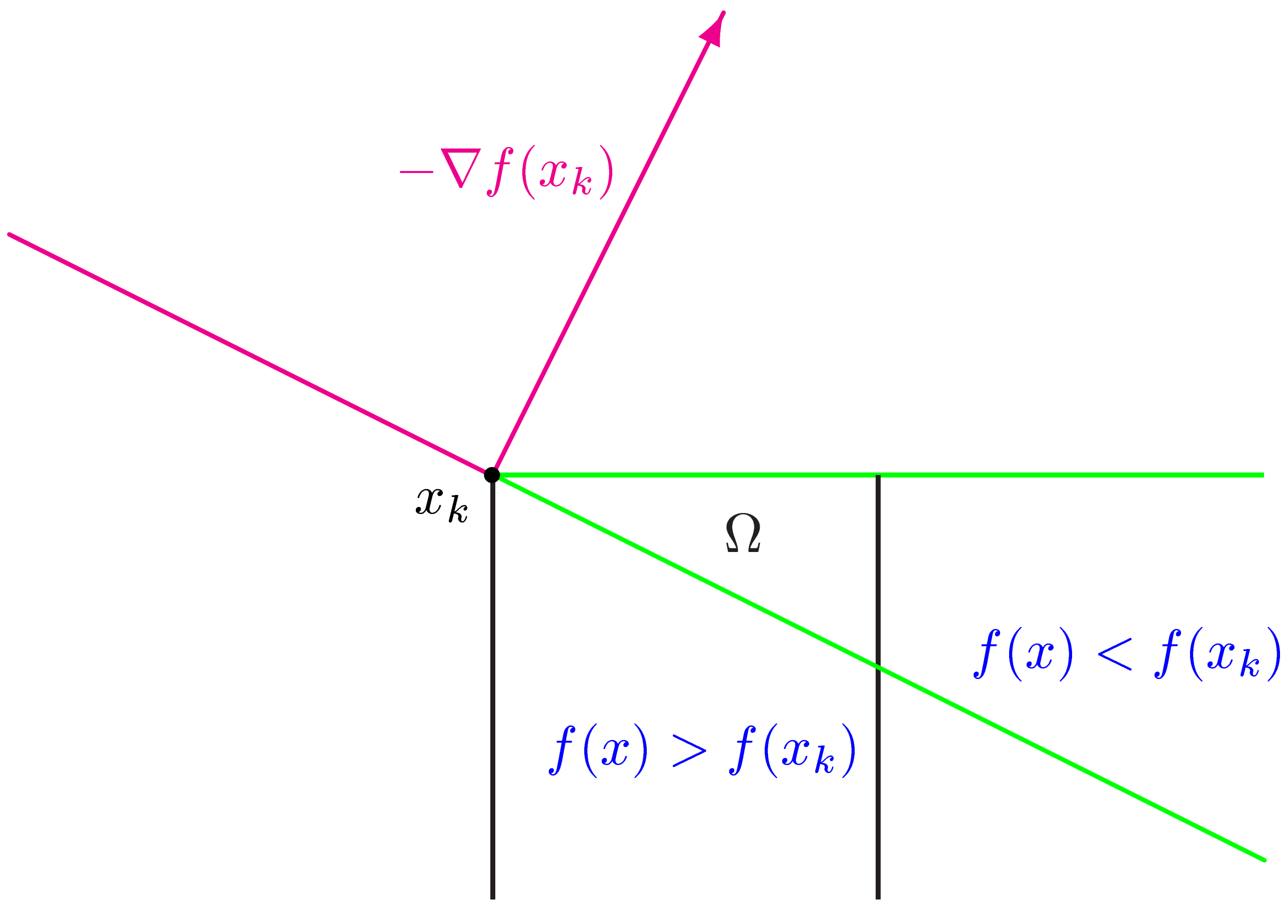
1. A *sufficient* set of directions in the form of a *positive basis* (provided f is continuously differentiable).
2. Lattice prevents steps that are either too long or too short.

   Accept *any* step that stays on the lattice so long as $f(x_k + s_k) < f(x_k)$ (i.e., *simple* decrease).

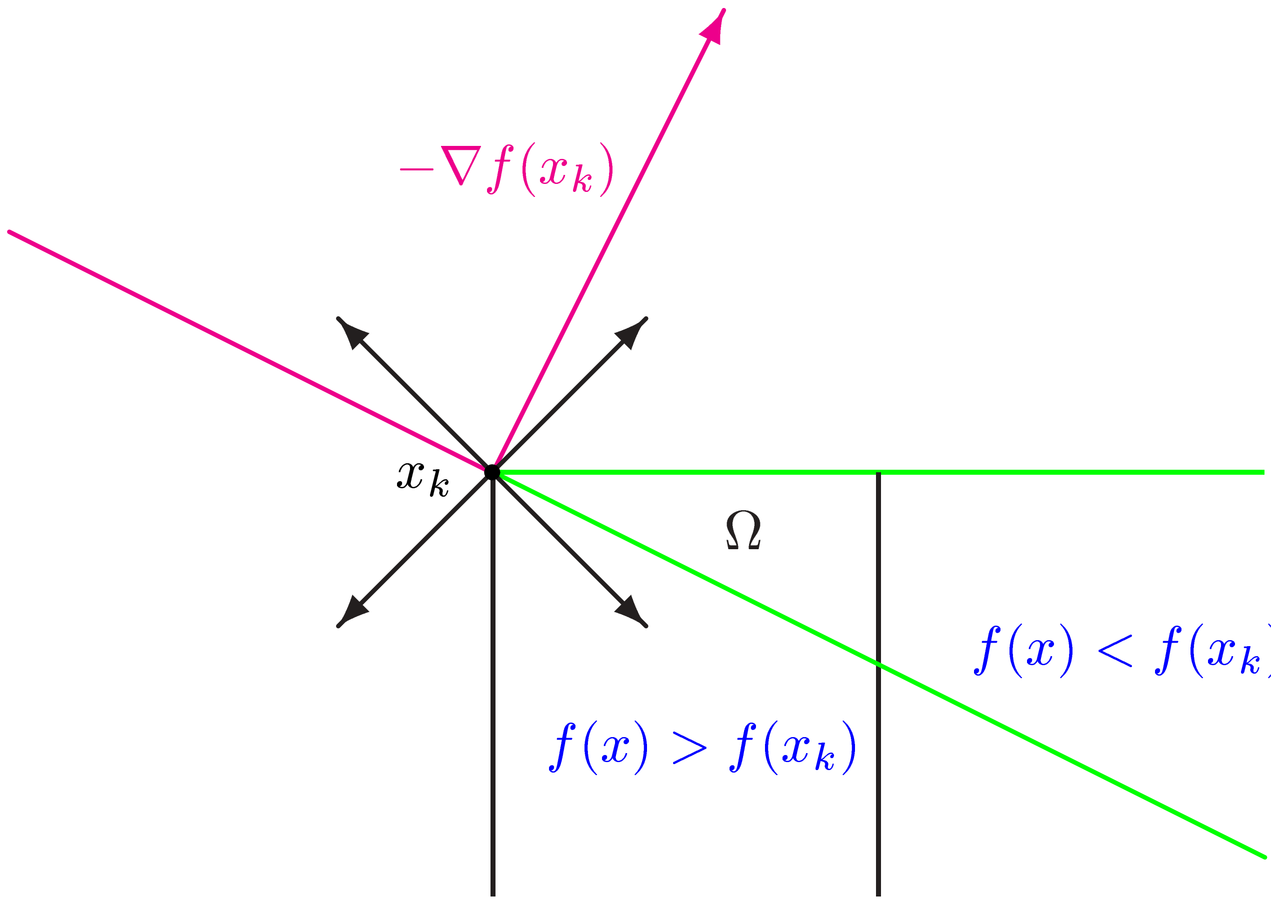
What Happens When We Introduce Bound Constraints?



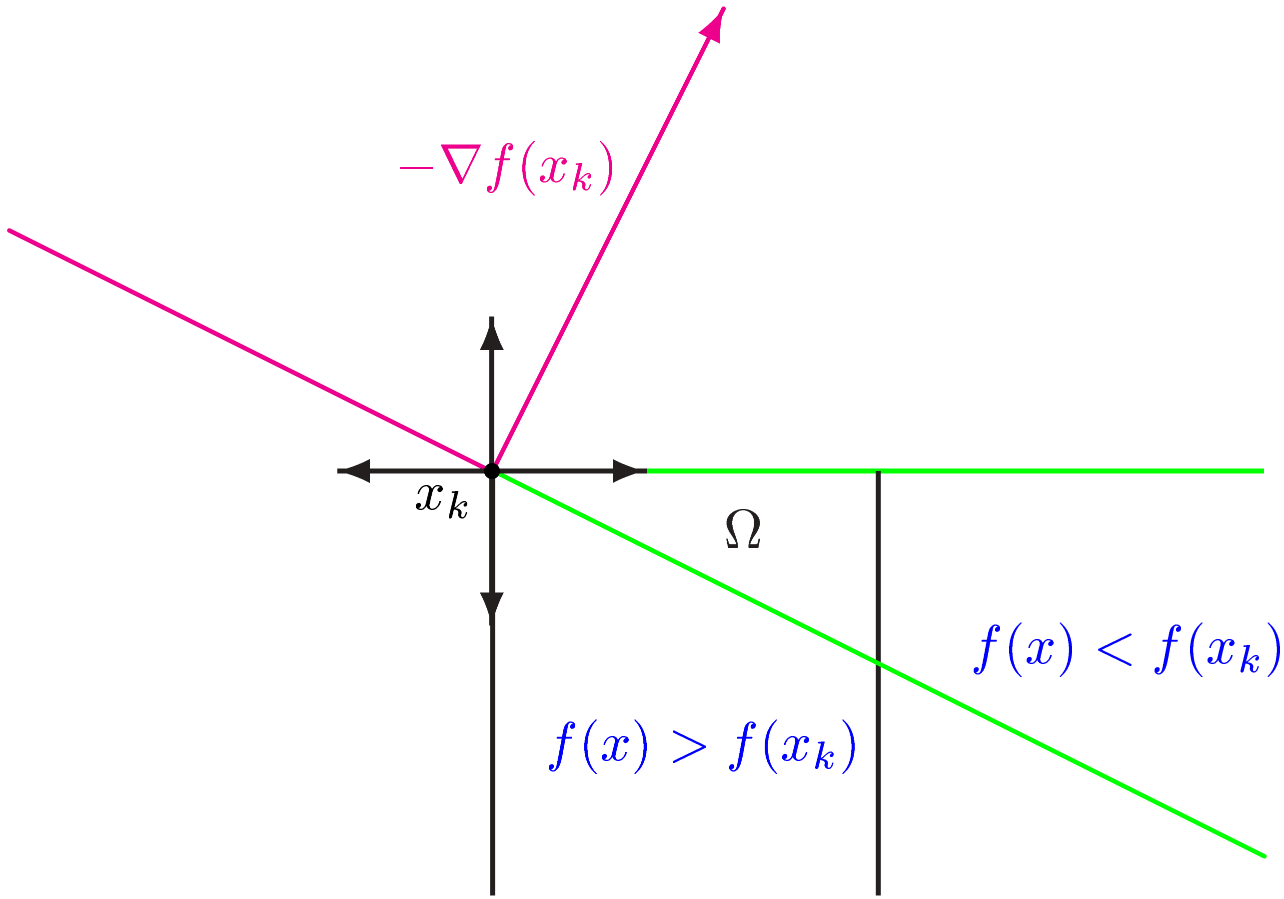
Cone of Feasible Descent Directions



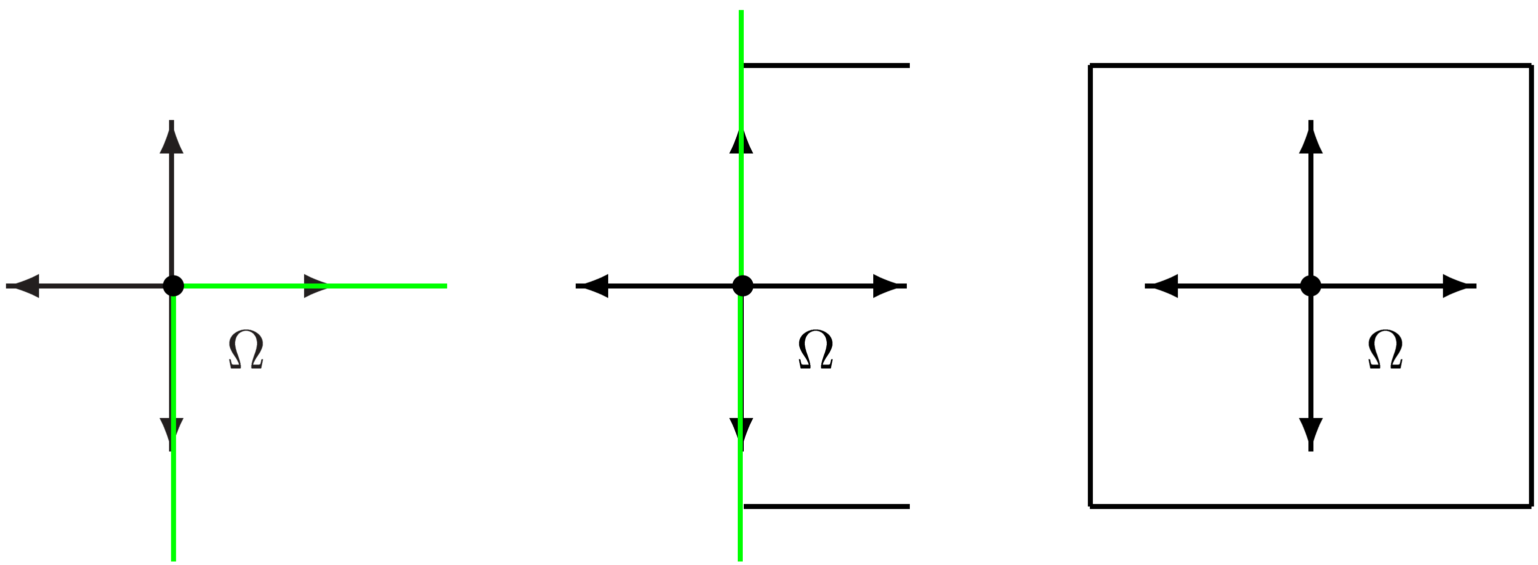
Using Some Positive Basis Is No Longer Sufficient



Pattern Must Conform with the Geometry of Nearby Constraints



Easy for Bound Constraints



A Single Pattern for Bound Constraints

The set of coordinate directions $\mathcal{D} = \{e_1, -e_1, \ldots, e_n, -e_n\}$ suffices.

1. The set $\mathcal{D}$ forms a positive basis for $\mathbb{R}^n$.

   Guarantees a gradient-related direction.

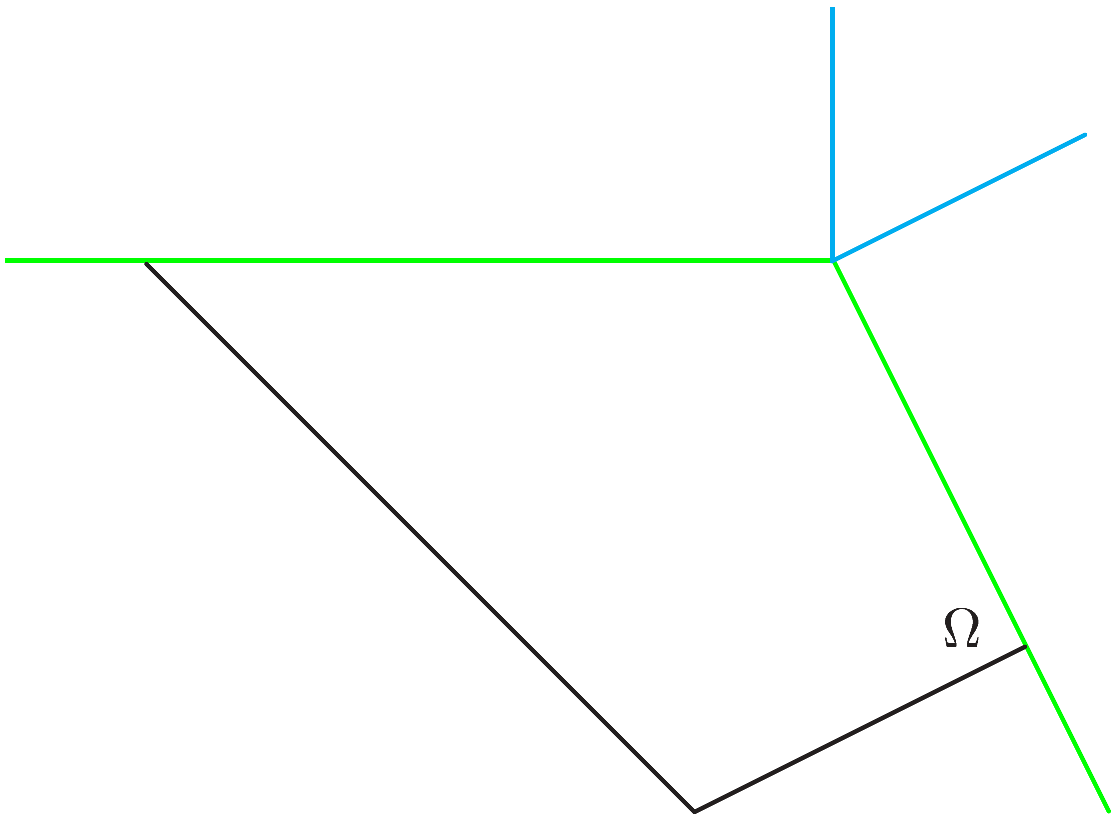
2. The set $\mathcal{D}$ always conforms to the geometry of the constraints.

   Guarantees a direction in the cone of feasible directions.

Straightforward to preserve the lattice structure and thus protect the step length and preserve the step acceptance criterion.

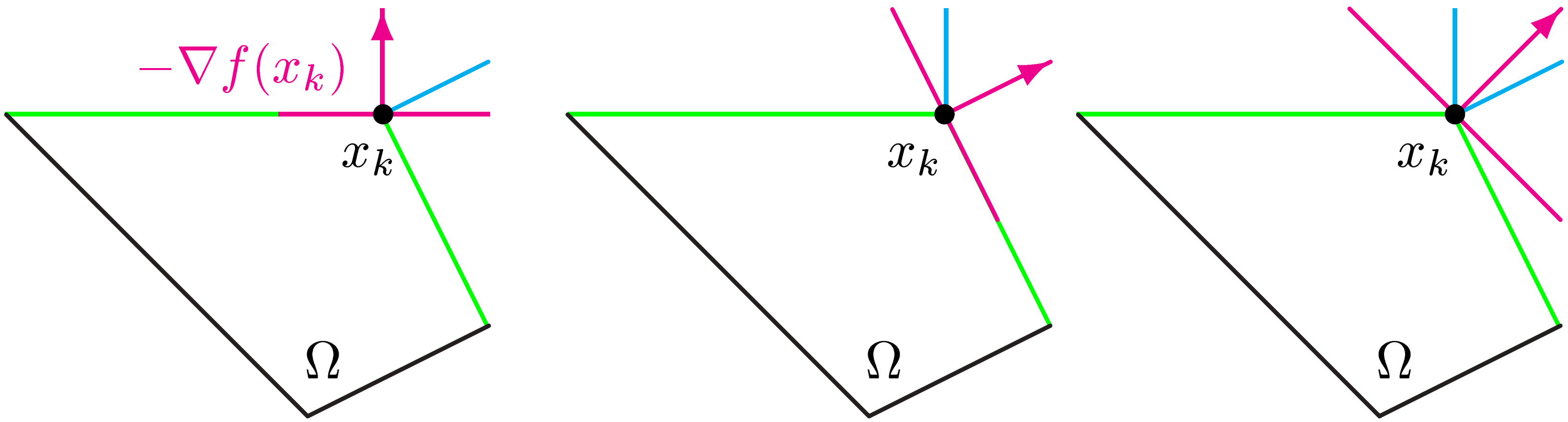
Drawback: Lose flexibility in the definition of the set of search directions.

What Happens When We Introduce Linear Constraints?



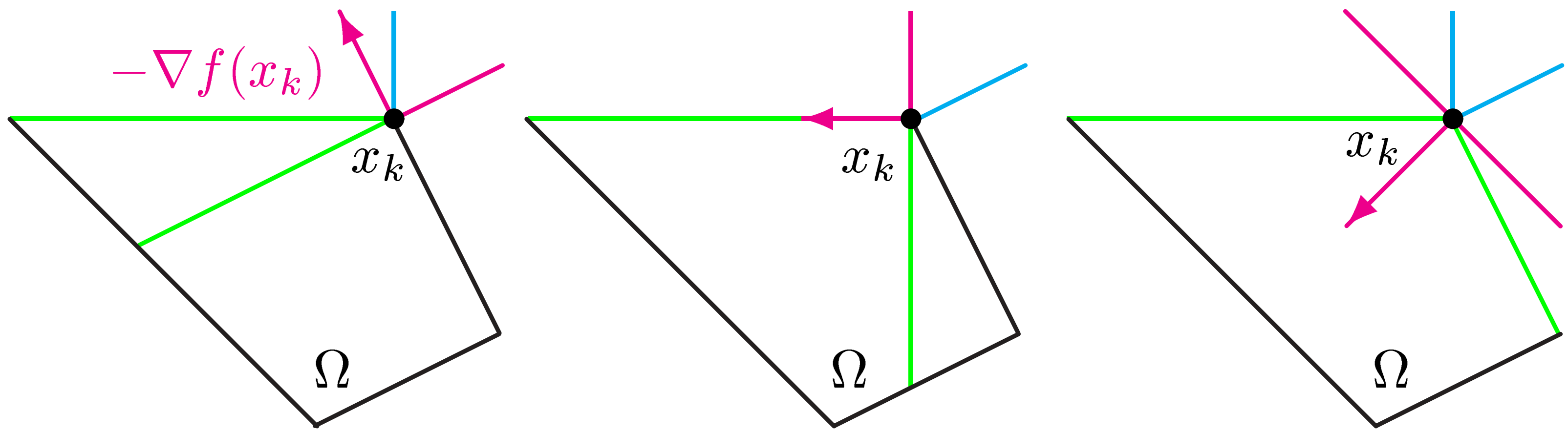
Geometry is no longer as simple as it was for bound constraints.

Optimality



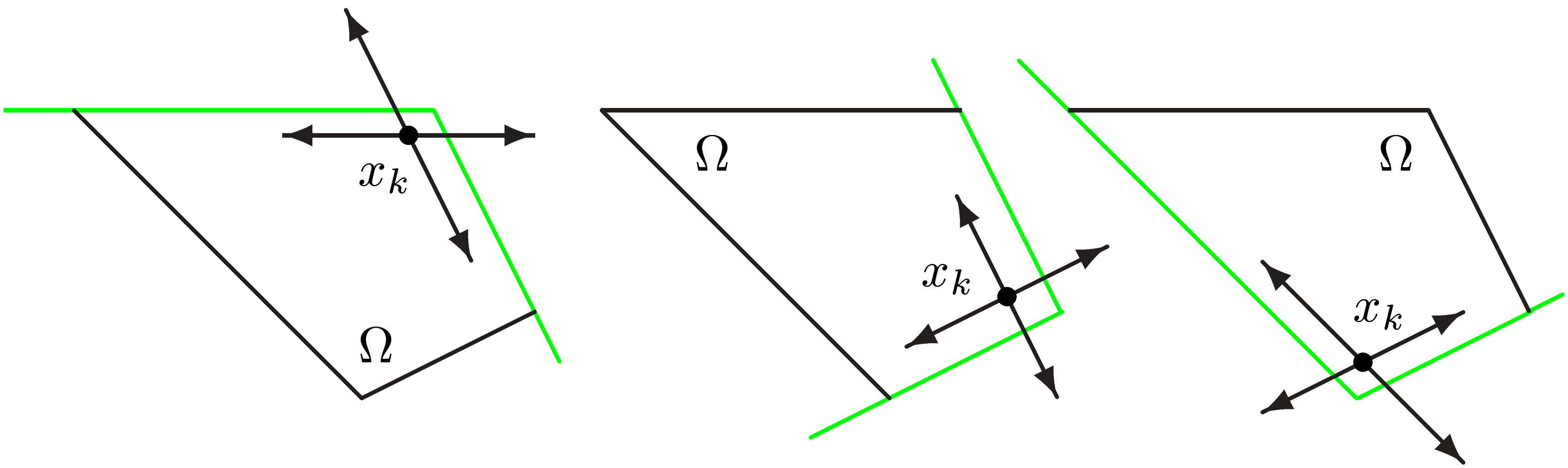
Standard for *any* optimization algorithm.

Cones of Feasible Descent Directions



Standard for *any* optimization algorithm.

Pattern Search Directions Depend on Nearby Constraints



Sufficient set if mirrors geometry of nearby constraints.

How to Determine "Good" Directions

Use the constraint matrix A (from $\ell \leq Ax \leq u$) to generate the search directions.

Two choices:

1. Enumerate all possible search directions by processing all possibilities before initiating the search.

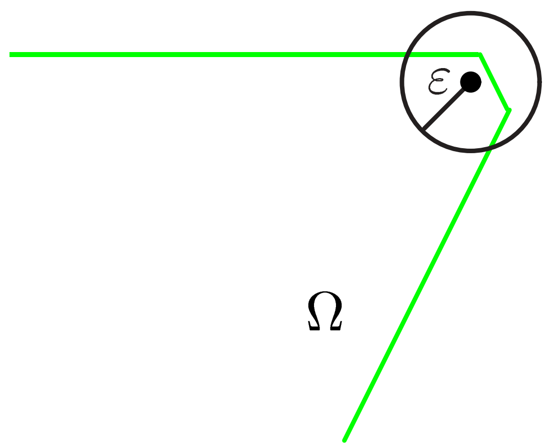
   Combinatorial growth. Related to the combinatorial problem of vertex enumeration.

2. Dynamically generate only those search directions needed to conform to the nearby constraints.

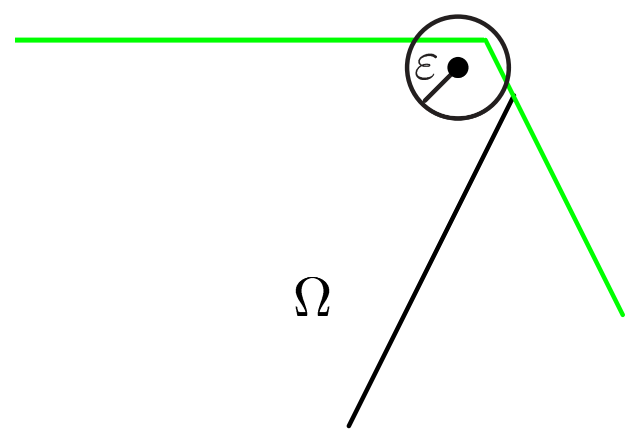
   A sort of "active set" strategy related to Dantzig's simplex method for linear programming.

   What is "nearby"?

Straightforward to Define "Nearby" When Constraints Are Nondegenerate



Oops! Not a cone.



Reduce ε until there are at most n constraints nearby.

Complication When Constraints Are Degenerate

Number of constraints that determines a vertex is *greater* than n.

Example in $\mathbb{R}^3$:

a feasible region defined by a *box* (nondegenerate)

every vertex defined by exactly three constraints

—versus—

a feasible region defined by a *pyramid* (degenerate)

one vertex defined by four constraints

Open question: How to *correctly* and *efficiently* identify the cone of feasible directions?

Active Research Questions

1. Adaptively generating the correct search directions for linear constraints.
2. Developing effective asynchronous parallel strategies for handling linear constraints.

Future Research Questions

1. General (nonlinear) constraints—including effective asynchronous parallel strategies
2. Effectiveness of alternate step acceptance criteria
   - sufficient decrease [Lucidi/Sciandrone, 1997; Garcia–Palomares/Rodriguez, 1999(?)]

   Imposing stronger acceptance criteria allows us to relax the lattice restriction.
3. Feasible versus infeasible iterates approaches.

Thank You

To Sandia National Laboratories, and the CSRI in particular, for providing support.

To the Computational Sciences and Mathematics Research Department, Org. 8950, Livermore, for providing me a (relatively) quiet work place and good companionship.

I have had both an enjoyable and a productive visit.